1. Derive the update rule and show how to train a 3-layer (1 input layer, 1 hidden layer, and 1 output layer) network with backpropagation for regression using the Mean Square Error loss. Assume that you are using the Sigmoid activation function for the hidden layer. Explain briefly how this is different from the update rule for the network trained for binary classification using log loss.

2. For the given data in train.csv and test.csv, construct a neural network for the regression task. Your network must have 1 input layer, 1 hidden layer, and 1 output layer. Use sigmoid to be your activation function for the hidden layer. Remember: since it is a regression problem, you should use a linear activation for the final (output) layer. The number of neurons in your output layer should be 1 since you are predicting only one output value.

The data is already split to have your input data for training (X\_train.csv) and testing (X\_train.csv) and their corresponding target valuesY\_train.csv and Y\_test.csv, respectively.

You can load the data as follows: X\_train = np.loadtxt("X\_train.csv")

Implement the backpropagation algorithm and train your network until convergence.

Answer the following questions:

1. Report the average MSE loss and the accuracy.

2. Plot the loss and accuracy as a function of the number of iterations.

3. What is the effect of the learning rate on the training process? Vary the learning rate to be between 0.001and 1.0 and plot the resulting accuracy as a function of learning rate.

4. What is the effect of the number of neurons in the hidden layer? Vary the number of neurons from 1to 10 and report the final loss and accuracy along with a brief description (2-3 lines) of your observation.

5. What is the effect of the activation functions in the network? Explore two different activation functions other than sigmoid such as tanh, linear,orReLU.

Hint: Update rules for weights will change with a change in activation function.

Back propagation algorithm is the most important building block in a neural network. The algorithm is used to train a neural network using chain rule, which works in such a way, that after every pass through a network, a backward pass is performed while considering the model parameters (weights and biases).

A 1 layer neural network contains 1 neuron in the input layer, 1 neuron in the hidden layer and 1 neuron in the output layer. A 4 layer neural network contains 4 neurons in the input layer, 4 neurons in the hidden layer and 1 neuron in the output layer.

**Input Layer:**

The data in the input layer is represented using scalars i.e. 1,2,3,4.

**Hidden Layer:**

The final values of the hidden neurons in the hidden layer are calculated using z^ l, i.e. weighted inputs in the layer l and a^l, activations in layer l. For layer 2 and 3 of a 4 layer neural network, the equations can be written as

**z(2)  = W(1)x + b(1)**

**a(2) = f(z(2) )**

For Layer 3, the equations are

**z(3)  = W(2)x + b(2)**

**a(3) = f(z(3) )**

Activations, a(2) and a(3), are calculated using the activation function f, as this function is mostly non linear i.e. sigmoid, relu, tanh, so it allows the network to learn, the complex patterns in the data

Let us for example consider layer3 and its parameters as an example.

W2 is the weight matrix of shape (n, m) where n represents the number of output neurons (neurons in the next layer) and m represent, the number of input neurons (neurons in the previous layer), if we have n = 2 and m =4, then we have a matrix of 2 rows and 4 columns

x is the input vector of shape (m, 1) where m is the number of input neurons, in this case, so considering that a 4 layer neural network will have 4 neurons in the input layer so size will be 4 x 1

b1 is the bias of the vector of shape (n, 1), where n represents, the number of neurons in the current layer. In layer 2, of a 4 layer neural network we have n = 2, so we will have a matrix of size 2 x1.

**Output Layer**

In the final part, we have an output layer, which produces the predicted value

**s = W(3)a(3)**

The main part in the forward propagation is to evaluate the predicted output with the expected output, this is done through cost function which can be done simple using Mean-Squared Error.

The cost function is represented as

**C = cost(s,y)**

Our main aim is to minimize the cost function and to bring the predicted value near to the expected value through doing back propagation by adjusting the weights and biases. The level of adjustment is determined through gradient of the cost function with respect to those parameters.

The derivative of C measures the change in the output value with respect to the change in the input value.

The further computation of these gradient is done through the chain rule.